Джерела:

1. [Firestore  |  Firebase (google.com)](https://firebase.google.com/docs/firestore)
2. [Firebase Authentication (google.com)](https://firebase.google.com/docs/auth)
3. [Firebase Authentication Password Hashing (firebaseopensource.com)](https://firebaseopensource.com/projects/firebase/scrypt/)
4. <https://www.ibm.com/topics/2fa>
5. [Introduction to the Admin Auth API  |  Firebase Authentication (google.com)](https://firebase.google.com/docs/auth/admin/)
6. <https://react.dev/>
7. <https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started>
8. <https://reactrouter.com/>
9. [Firebase Authentication (google.com)](https://firebase.google.com/docs/auth/)
10. <https://dev.mysql.com/doc/>
11. <https://www.postgresql.org/docs/>
12. <https://docs.mongodb.com/manual/>
13. <https://cloud.google.com/docs/>
14. <https://docs.aws.amazon.com/>
15. <https://docs.microsoft.com/en-us/azure/>
16. <https://www.geeksforgeeks.org/firestore-and-its-advantages/>
17. [What is Cloud Firestore? (back4app.com)](https://blog.back4app.com/what-is-cloud-firestore/)

### Аналіз відомих алгоритмічних та технічних рішень

Реєстрація користувачів – ключовий етап створення будь-якої системи, і вибір правильного методу грає критичну роль у забезпеченні безпеки, ефективності та зручності для кінцевого користувача. В ряду методів реєстрації, від керування власною системою до використання сторонніх сервісів, можна виділити кілька підходів.

Існує кілька різних методів розв'язання задачі реєстрації користувачів. Найпоширенішими з них є:

* Використання локальної бази даних;
* Використання сторонньої служби аутентифікації.

Використання локальної бази даних дозволяє зберегти облікові записи користувачів в локальній базі даних додатка. Цей метод є простим у реалізації, але він має ряд недоліків. Зокрема, він не дозволяє масштабувати додаток, оскільки база даних може бути перевантажена при великій кількості користувачів.

Використання сторонньої служби аутентифікації, в свою чергу, зберігає облікові записи користувачів в сторонній службі аутентифікації, наприклад, Google, Facebook або Twitter. Цей метод дозволяє масштабувати додаток, оскільки навантаження розподіляється між різними серверами сторонньої служби.

Firebase Auth є сторонньою службою аутентифікації, яка надається компанією Google. Firebase Auth пропонує широкий спектр функцій, які дозволяють легко реалізувати реєстрацію користувачів у веб-додатках. Серед переваг Firebase Auth можна визначити:

Firebase Auth має декілька переваг:

* Економія часу на розробку методів аутентифікації;
* Отримання детальної аналітики та демографічної інформації про користувачів;
* Інтеграція з іншими сервісами Firebase, що дозволяє легко використовувати сервіси разом;
* Використання стандартів, таких як OAuth 2.0 та OpenID Connect, що може бути легко інтегрований з власним бекендом[9];
* Захист данів користувачів через надання безпечних методів аутентифікації.

Задача збереження даних користувачів є однією з найважливіших задач у веб-додатках. В нашому випадку ми потребуємо зберігати дані про користувачів, такі як: ім’я, пошта, дата реєстрації, список збережених фільмів дозволяє зберігати інформацію про користувачів наприклад, їхні імена, паролі, адреси електронної пошти, коментарі та інші дані.

Існує кілька різних методів розв'язання задачі збереження даних користувачів. Найпоширенішими з них є:

* Використання локальної бази даних;
* Використання сторонньої бази даних;
* Використання хмарної бази даних.

Локальна база даних дозволяє зберігати дані користувачів в локальній базі даних додатка. Цей метод є простим у реалізації, але він має ряд недоліків. Зокрема, він не дозволяє масштабувати додаток, оскільки база даних може бути перевантажена при великій кількості користувачів.

При методі використання сторонньої БД дані користувачів зберігаються в сторонній базі даних (MySQL[10], PostgreSQL[11] або MongoDB[12]). Цей метод вже дозволяє масштабування, оскільки навантаження розподіляється між різними серверами сторонньої бази даних.

Хмарні БД дозволяють зберігати усі необхідні дані у хмарі (Google Cloud Platform[13], Amazon Web Services[14] або Microsoft Azure[15]). Вони використовуються для структурованого утримання інформації та надають потужні можливості запитів для отримання та маніпулювання даними. Через їх зручність, надійність та функціонал вони мають перевагу над іншими методами у використанні.

Firestore є хмарною базою даних NoSQL, яка надається компанією Google. Вона надає гнучке, масштабоване зберігання даних для мобільних, веб- та серверних додатків. Серед найважливіших переваг інструменту можна зазначити наступні[16]:

* Асинхронні запити;
* Взаємодія із БД через прості запити, за допомогою яких можна виконувати складні запити NoSQL;
* Надання синхронізації в режимі офлайн, яка допомагає користувачам взаємодіяти з додатком, коли немає підключення до мережі;
* Автоматичне горизонтальне масштабування відповідно до навантаження[17];
* Використання протоколу HTTPS для шифрування передачі даних та вбудовані механізми безпеки, що визначають правила захисту від несанкціонованого доступу.

Іншою проблемою є наповнення контентом нашого застосунку. Оскільки бібліотека фільмів має бути різноманітною та актуальною, маємо вирішити спосіб отримання даних про фільми.

Одним з методів є власноручне додавання фільмів. В цьому випадку, адміністратори будуть вручну додавати та оновлювати інформацію про фільми. Цей метод часом використовується в невеликих проектах, проте він потребує багато часу та є неефективний для великих обсягів даних. Отже, необхідно отримувати інформацію з окремих публічних джерел.

Наш застосунок буде використовувати TMDB

2.1.3 Взаємодія з API:

Для отримання інформації про фільми використовується API від tmdb. У цьому випадку, алгоритми обробки запитів та обробки отриманої інформації оптимізовані для ефективного використання ресурсів сервера та швидкої відповіді на запити.

2.2 Технічні рішення

2.2.1 Axios для взаємодії з API:

Бібліотека Axios використовується для здійснення HTTP-запитів, що дозволяє ефективно взаємодіяти з API tmdb. Axios надає зручний та функціональний інтерфейс для виконання різних видів запитів та обробки відповідей.

2.2.2 Firebase для аутентифікації та збереження даних:

Використання Firebase Auth та Firestore надає готові рішення для реєстрації та аутентифікації користувачів, а також для збереження та отримання даних з бази даних.

2.2.3 React для фронтенду:

React обрано в якості основної бібліотеки для розробки фронтенду, оскільки він забезпечує швидку реакцію інтерфейсу користувача на зміни, легку роботу з компонентами та ефективне управління станом додатку.

<У пункті викладають:

відомі алгоритми для розв’язання задач чи підзадач у вашій розробці, в кінці робиться порівняльний аналіз алгоритмів та обирається той, який ви будете використовувати у розробці. Або робиться висновок про необхідність розробки оригінальних алгоритмів чи модифікацію існуючих. Розглядаються відомі технічні рішення, які допоможуть у реалізації розробки: архітектурні паттерни, платформи, тощо. Для технічних рішень також робиться порівняльний аналіз та обираються ті, що будуть використані у розробці. Можливо зробити висновок про розробку оригінального технічного рішення, чи модифікацію існуючого.>

### Аналіз допоміжних програмних засобів та засобів розробки

<У пункті викладають:

опис допоміжних програмних засобів та їх порівняльний аналіз, якщо існує кілька аналогів (мова йде про сторонні бібліотеки, пакети, фреймворки, тощо). Розглядаються засоби розробки за допомогою яких можна виконати вашу розробку, робиться їх порівняльний аналіз та на його основі обираються ті, що вам підходять (маються на увазі мови програмування та IDE).>

### Аналіз відомих програмних продуктів

<У пункті викладають:

опис готових програмних продуктів за вашою предметною областю, які частково чи повністю реалізують функціонал описаний у технічному завданні. Обов’язково має бути порівняльна таблиця функціоналу та особливостей з вашою розробкою.>

Для порівняння курсової роботи з аналогом можна скористатись таблицею 1.3.

Таблиця 1.3 – Порівняння з аналогом

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Функціонал | Дипломний проєкт(назва) | Аналог для порівняння 1 | Аналог для порівняння n | Пояснення |
|  |  |  |  |  |

## Аналіз вимог до програмного забезпечення

<У підрозділі викладають:

опис головного(-х) функціоналу розробки, діаграму варіантів використання та опис варіантів використання. Обсяг 5 стр., чи більше >

Головною функцією програмного забезпечення є …, більше функцій можна побачити на рисунку 1.3.

Рисунок 1.3 – Діаграма варіантів використання

В таблицях 1.2 - 1.17 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 - Варіант використання UC-1

|  |  |
| --- | --- |
| Use case name | Реєстрація користувача |
| Use case ID | UC-01 |
| Goals | Реєстрація нового користувача в системі |
| Actors | Гість (незареєстрований користувач) |
| Trigger | Користувач бажає зареєструватися |
| Pre-conditions | - |
| Flow of Events | Користувач переходить на сторінку реєстрації. В поля для реєстрації вводяться відповідні дані: пошта користувача, пароль в системі, та його повтор для підтвердження, а також чек бокс для підтвердження умов сервісу. Після заповнення даних користувача натискає кнопку реєстрації. Після цього з’являється повідомлення про успішну реєстрацію, і користувач перенаправляється на сторінку входу. |
| Extension | В випадку введення не коректних даних, кнопка реєстрації стає неактивною. Якщо якесь конкретне поле введено некоректно, то воно підсвічується червоним надписом. |
| Post-Condition | Створення сторінки користувача, перехід на сторінку входу |

### Розроблення функціональних вимог

<У пункті викладають:

модель вимог, їх опис та матрицю трасування.>

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. На рисунку 1.4 наведено загальну модель вимог, а в таблицях 1.18 – 1.27 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 1.5.

Рисунок 1.4 – Модель вимог у загальному вигляді

Таблиця 1.16 – Функціональна вимога FR-1

|  |  |
| --- | --- |
| Назва | Реєстрація користувача |
| Опис | Система повинна надавати можливість реєстрації користувачеві шляхом введення пошти, паролю, підтвердження паролю. |

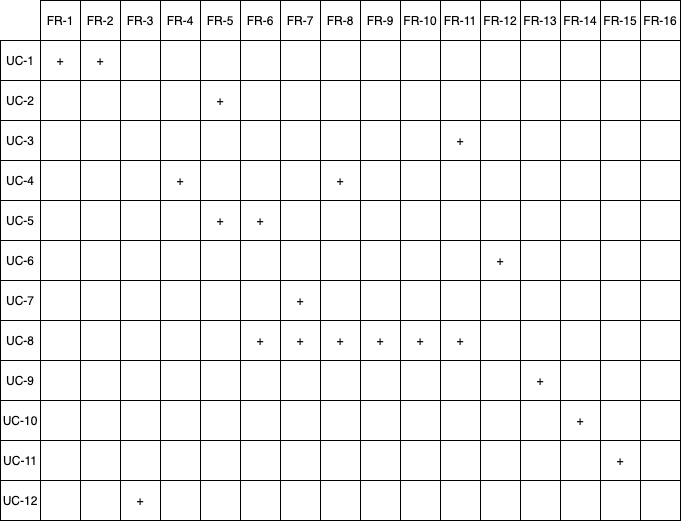


Рисунок 1.11 – Матриця трасування вимог

### Розроблення нефункціональних вимог

<У пункті викладають:

опис нефункціональних вимог.>

## Постановка задачі

<У підрозділі викладають:

що саме ви збираєтесь реалізувати. Це не зовсім вимоги і варіанти використання. Це достатньо узагальнене поняття з метою, цілями та задачами, що підлягають розв’язанню у результаті розробки програмного забезпечення на курсову роботу. Обсяг 1 сторінка, чи більше. >

## Висновки до розділу

< Необхідно стисло описати усе, що було виконано у даному розділі. Обсяг 0,75-1 сторінка>

# МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Моделювання та аналіз програмного забезпечення

<У підрозділі викладають:

опис бізнес процесів вашої розробки та їх наочне представлення за допомогою засобів моделювання (BPMN та інші). Обсяг 2 сторінки, чи більше. >

Для опису бізнес процесу програмного забезпечення використовується BPMN модель (рисунок 2.1).

Опис послідовності створення облікового запису користувача:

* користувач переходить на сторінку реєстрації;
* користувач заповнює поля реєстрації;
* якщо введені поля, не відповідають шаблону заповнення на клієнтській стороні, відповідні поля підсвічуються помилкою;

## Архітектура програмного забезпечення

<У підрозділі викладають:

втілення архітектурного паттерну для вашого ПЗ та деталізація його компонентів за необхідності. Необхідно виконати повний опис архітектури (схеми, таблиці та ін.). Обсяг 2 сторінки, чи більше.>

## Конструювання програмного забезпечення

<У підрозділі викладають:

опис оригінальних алгоритмів чи модифікацій існуючих. Опис структур даних, програмних структур та ін. Опис бази даних з представленням концептуальної, логічної чи фізичної моделі та з описом сутностей чи таблиць. Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці. Аналіз системних вимог. Обсяг 4 сторінки, чи більше.>

Бібліотека xpring, яка виконує запити в мережу блокчейн – однопоточна, тобто не може оброблювати паралельні запити. Через таку специфіку бібліотеки, довелось розробляти рішення яке б не блокувало запити з різних потоків, або інших частин коду. За основу ідеї алгоритму була взята багатопоточність самої мови програмування java. Java для вирішення проблем з потоками представляє ділянки коду які можна синхронізувати, що унеможливлює одночасний доступ до нього з різних потоків. Ця синхронізація проходить за допомогою передачі управління об’єкта-монітора. Алгоритм вирішення проблеми з доступом до однопоточної бібліотеки наведено на рисунку 2.3.

В якості системи управління базами даних використовується Postgres. База даних серверу призначена для зберігання користувачів, а також даних про їх …. Опис таблиць бази даних наведено у таблицях 2.11 - 2.14. Модель бази даних наведена на рисунку 2.12.

Таблиця 2.11 – Опис таблиці user

|  |  |  |  |
| --- | --- | --- | --- |
| Таблиця | Назва поля | Тип даних | Опис |
| user | id | serial | ідентифікаційний номер користувача |
| email | varchar | електронна пошта користувача |
| password\_id | int | посилання на запис у таблиці password, де зберігається пароль користувача |

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 2.22.

Таблиця 2.22 – Опис утиліт

|  |  |  |
| --- | --- | --- |
| №  п/п | Назва утиліти | Опис застосування |
| 1 | IntelliJ IDEA | Головне середовище розробки програмного забезпечення серверної частини курсової роботи. |
| 2 | Postman | Програмне забезпечення необхідне для тестування rest запитів. Використовувалось для тестування API інтерфейсів, та клієнтських запитів. |
| 3 | MySQL Workbench | Програмне забезпечення яке надає легкий графічний інтерфейс для доступу до бази даних. |

## Аналіз безпеки даних

<У підрозділі викладають:

аналіз вразливостей ПЗ та будь-які питання пов’язані з безпекою даних. >

## Висновки до розділу

< Необхідно стисло описати усе, що було виконано у даному розділі. Обсяг 0,75-1 сторінка>

# АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Аналіз якості ПЗ

<У підрозділі викладають:

аналіз якості ПЗ за певними метриками. Обсяг 2 сторінки, чи більше.>

## Опис процесів тестування

<У підрозділі викладають:

опис процесів тестування та приклади тестів. Обсяг 4 сторінки, чи більше.>

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.3 – 3.30.

Таблиця 3.3 – Тест 1.1

|  |  |
| --- | --- |
| Тест | Реєстрація користувача |
| Модуль | Реєстрація користувача |
| Номер тесту | 1.1 |
| Початковий стан системи | Користувач знаходиться на сторінці реєстрації |
| Вхідні данні | Електронна пошта, пароль, підтвердження паролю |
| Опис проведення тесту | У відповідні поля вводяться: коректна електронна пошта, яка до цього не була зареєстрована в системі, пароль від 10 до 64 символів, який містить хоча б з одну англійську літеру, одне число і один спеціальний символ, і який не входить у топ 10000 найпопулярніших паролей, підтвердження паролю, яке співпадає з раніше введеним паролем. Після цього … |
| Очікуваний результат | Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації. |
| Фактичний результат | Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації. |

## Опис контрольного прикладу

<У підрозділі викладають:

повний опис контрольного прикладу з усіма можливими розгалуженнями та особливостями. Кроки доповнюють ілюстраціями. Не обов’язковий розділ.>

## Висновки до розділу

< Необхідно стисло описати усе, що було виконано у даному розділі. Обсяг 0,75-1 сторінка>

# ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Розгортання програмного забезпечення

<У підрозділі викладають:

повний опис покрокового розгортання ПЗ. Кроки доповнюють ілюстраціями. Обсяг 2 сторінки, чи більше.>

Клієнтську і серверну частини програмного забезпечення було вирішено розгорнути на платформі Heroku. Для розгортання було використано сервіс GitHub Actions, який надає можливості для постійної інтеграції і розгортання [13].

Розгортання починається коли новий код застосунку доставляється у репозиторій у гілку main. Тоді у середовищі GitHub Actions створюється Docker image за допомогою Dockerfile, що знаходиться у проекті. Цей image розгортається у Heroku за допомогою пакету heroku-deploy. Інформацію про розгортання клієнтської і серверної частини проекту можна побачити на рисунках 4.1 і 4.2.

Рисунок 4.1 - Інформація про розгортання клієнту

## Підтримка програмного забезпечення

<У підрозділі викладають:

опис того, як буде виконуватись підтримка програмного забезпечення. Ілюстрації. Обсяг 1 сторінка, чи більше.>

Користувачі повинні мати можливість отримати нову версію консольного застосунку з кожною версією. До того ж кожна нова версія консольного застосунку повинна бути опублікована в npm. Для автоматизації цього процесу був використаний сервіс GitHub Actions.

Створення нового випуску починається, коли нова версія консольного застосунку доставляється у репозиторій у гілку main, тобто коли commit має tag формату “v\*.\*.\*.”, де замість “\*” знаходиться число. Тоді у середовищі GitHub Actions встановлюється NodeJS. Після цього для проекту встановлюються залежності і проект збирається. Bash скрипт за допомогою бібліотеки pkg генерує виконувані файли (executables) для Linux і для Windows, та пакує файл для Linux у .deb пакет. Після цього .deb пакет і файл для Windows архівуються,

## Висновки до розділу

< Необхідно стисло описати усе, що було виконано у даному розділі. Обсяг 0,75-1 сторінка>

# ВИСНОВКИ

У висновках викладають найважливіші наукові й практичні результати роботи та наводять:

* оцінку одержаних результатів і їх відповідність сучасному рівню наукових і технічних знань;
* ступінь впровадження та можливі галузі або сфери використання результатів роботи;
* наукову, науково-технічну, соціально-економічну значущість роботи;
* доцільність продовження досліджень за відповідною тематикою тощо.

Також у висновках необхідно відобразити стан вирішення усіх поставлених в курсовій роботі задач.

В результаті виконання курсової роботи було спроєктовано …

В якості середовища розробки обрано …

У якості БД використано …

Після реалізації застосунку він був протестований на пристроях з різними версіями Android, з різними розмірами екранів щоб переконатися, що додаток акуратно відображається на різних пристроях.

Наукова новизна роботи (якщо вона є) полягає в наступному (достатньо вказати щось одне).

Вперше:

* реалізовано можливість запитів від пацієнта до лікаря;
* використано те-то, що дозволило те-то.

Модифіковано:

* те-то, що дозволило те-то.

Набуло подальший розвиток:

* те-то, що дозволило те-то.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. [Оформлення бібліографії ДСТУ 7.1:2006](https://drive.google.com/file/d/1VUr7fwKHOk8t1u8sh-Sv2dlEc_6OBoIS/view?usp=sharing)
2. [Оформлення бібліографії ДСТУ 8302-2015](https://drive.google.com/file/d/17RYEYVnSXolL0S8D1k85VfReYd2yC6WO/view?usp=sharing) з прикладами